CRYPTANALYSIS OF THE ALGEBRAIC ERASER AND SHORT EXPRESSIONS OF PERMUTATIONS AS PRODUCTS

ARKADIUS KALKA, MINA TEICHER, AND BOAZ TSABAN

Abstract. On March 2004, Anshel, Anshel, Goldfeld, and Lemieux introduced the *Algebraic Eraser* scheme for key agreement over an insecure channel, using a novel hybrid of infinite and finite noncommutative groups. They also introduced the *Colored Burau Key Agreement Protocol (CBKAP)*, a concrete realization of this scheme. CBKAP resisted cryptanalysis for four years.

We present general, efficient algorithms, which extract the shared key out of the public information provided by CBKAP. These algorithms are successful for all sizes of the security parameters, assuming that the keys are chosen with standard distributions.

Our methods come from probabilistic group theory, and have not been used before in cryptanalysis. In particular, we provide a simple and very efficient heuristic algorithm for finding short expressions of permutations as products of given random permutations. Our algorithm gives expressions of length $O(n^2 \log n)$, in time $O(n^4 \log n)$ and space $O(n^2 \log n)$, and is the first practical one for $n \geq 256$.

Remark. Algebraic Eraser is a trademark of SecureRF. The variant of CBKAP actually implemented by SecureRF uses proprietary distributions, and thus our results do not imply its vulnerability.

1. Introduction and overview

During the last decade, starting with the seminal papers [1, 10], attempts have been made to construct and analyze public key schemes based on noncommutative groups and combinatorial (or computational) group theory. The obvious motivation is that such systems may provide longer term security, and may (unlike the main present day public key schemes) be resistant to attacks by quantum computers. Moreover, these connections between combinatorial group theory and cryptography lead to mathematical questions not asked before, and consequently to new mathematical discoveries.

In this paper, we study a scheme falling in the above category, whose cryptanalysis leads to an algorithm with interest beyond the studied scheme.

The *Algebraic Eraser* key agreement scheme was introduced by Anshel, Anshel, Goldfeld, and Lemieux in the workshop *Algebraic Methods in Cryptography* held in Dortmund, Germany, on March 2004, and in the special session on *Algebraic Cryptography*, at the Joint International Meeting of the AMS, DMV, and ÖMG, held in Mainz, Germany, on June 2005. It was subsequently published as [2].

Apart from its mathematical novelty, the Algebraic Eraser has a surprisingly simple concrete realization, the *Colored Burau Key Agreement Protocol (CBKAP)*, which consists of an efficient combination of matrix multiplications, applications of permutations, and evaluations of polynomials at elements of a finite field. Despite its being brought to the attention of both the cryptography and the computational group theory communities, no weakness was identified in CBKAP in the four years which have passed since its introduction.

We present an efficient attack on this scheme, which recovers the shared key out of the public information, for all sizes of the security parameters. This attack was implemented and found successful on all of hundreds of instances, generated using standard distributions.

The methods, which make the attack applicable to large security parameters, come from probabilistic group theory, and deal with permutation groups. About half of the paper is dedicated to a new algorithm for finding short expressions of permutations as words in a given set of randomly chosen permutations. This algorithm solves efficiently instances which are intractable using previously known, provable or heuristic, techniques.

2. The Algebraic Eraser scheme

We describe here the general framework. The concrete realization will be described later.

2.1. **Notation, terminology, and conventions.** A *monoid* is a set M with a distinguished element $1 \in M$, equipped with an associative multiplication operation for which 1 acts as an identity. Readers not familiar with this notion may replace "monoid" with "group" everywhere, since this is the main case considered here.

Let G be a group acting on a monoid M on the left, that is, to each $g \in G$ and each $a \in M$, a unique element denoted ${}^{g}a \in M$ is assigned, such that:

(1) ${}^{1}a = a$;
(2) ${}^{gh}a = {}^{g}({}^{h}a)$; and
(3) ${}^{g}(ab) = {}^{g}a \cdot {}^{g}b$

for all $a, b \in M, g, h \in G$.

$M \times G$, with the operation

$$\big(a, g\big) \circ \big(b, h\big) = \big(a \cdot {}^{g}b, gh\big),$$

is a monoid denoted $M \rtimes G$.

Let N be a monoid, and $\varphi : M \to N$ a homomorphism. The *algebraic eraser* operation is the function $\star : (N \times G) \times (M \rtimes G) \to (N \times G)$ defined by

$$\big(a, g\big) \star \big(b, h\big) = \big(a\varphi({}^{g}b), gh\big). \tag{1}$$

The following identity holds for $\star$:

$$\big((a, g) \star (b, h)\big) \star \big(c, r\big) = \big(a, g\big) \star \big((b, h) \circ (c, r)\big) \tag{2}$$

for all $(a, g) \in N \times G, (b, h), (c, r) \in M \times G$.

Submonoids A, B of $M \rtimes G$ are *$\star$-commuting* if

$$\big(\varphi(a), g\big) \star \big(b, h\big) = \big(\varphi(b), h\big) \star \big(a, g\big) \tag{3}$$

for all $(a, g) \in A, (b, h) \in B$. In particular, if A, B $\star$-commute, then

$$\varphi(a)\varphi({}^{g}b) = \varphi(b)\varphi({}^{h}a)$$

for all $(a, g) \in A, (b, h) \in B$.

2.1.1. *Didactic convention.* Since the actions are superscripted, we try to minimize the use of subscripts. As a rule, whenever two parties, Alice and Bob, are involved, we try to use for Bob letters which are subsequent to the letters used for Alice (as is suggested by their names).

2.2. **The Algebraic Eraser Key Agreement Scheme.**

2.2.1. *Public information.*

(1) A positive integer m.
(2) $\star$-commuting submonoids A, B of $M \rtimes G$, each given in terms of a generating set of size k.
(3) Element-wise commuting submonoids C, D of N.

2.2.2. *The protocol.*

(1) Alice chooses $c \in C, (a_1, g_1), \dots, (a_m, g_m) \in A$, and sends
$$(p, g) = (c, 1) \star (a_1, g_1) \star \cdots \star (a_m, g_m) \in N \times G$$
(the $\star$-multiplication is carried out from left to right) to Bob.
(2) Bob chooses $d \in D, (b_1, h_1), \dots, (b_m, h_m) \in B$, and sends
$$(q, h) = (d, 1) \star (b_1, h_1) \star \cdots \star (b_m, h_m) \in N \times G$$
to Alice.
(3) Alice and Bob compute the shared key:
$$\begin{aligned} &(cq, h) \star (a_1, g_1) \star \cdots \star (a_m, g_m) = \\ &\quad = \ (dp, g) \star (b_1, h_1) \star \cdots \star (b_m, h_m). \end{aligned}$$
We will soon explain why this equality holds.

For the sake of mathematical analysis, it is more convenient to reformulate this protocol as follows. The public information remains the same. In the notation of Section 2.2.2, define

$$\begin{aligned} (a, g) &= (a_1, g_1) \circ \cdots \circ (a_m, g_m) \in A; \\ (b, h) &= (b_1, h_1) \circ \cdots \circ (b_m, h_m) \in B. \end{aligned}$$

By Equations (2) and (1), Alice and Bob transmit the information

$$\begin{aligned} (p, g) &= (c, 1) \star (a_1, g_1) \star \cdots \star (a_m, g_m) = (c, 1) \star (a, g) = (c\varphi(a), g); \\ (q, h) &= (d, 1) \star (b_1, h_1) \star \cdots \star (b_m, h_m) = (d, 1) \star (b, h) = (d\varphi(b), h). \end{aligned}$$

Using this and Equation (3), we see in the same manner that the shared key is

$$\begin{aligned}(cq,h)\star(a,g) &= (cq\varphi({}^{h}a),hg) = \\ &= (cd\varphi(b)\varphi({}^{h}a),hg) = (dc\varphi(a)\varphi({}^{g}b),gh) = \\ &= (dp\varphi({}^{g}b),gh) = (dp,g)\star(b,h).\end{aligned}$$

2.3. **When M is a group.** In the concrete examples for the Algebraic Eraser scheme, M is a group [2]. Consequently, $M \rtimes G$ is also a group, with inversion

$$(a,g)^{-1} = ({}^{g^{-1}}a^{-1}, g^{-1})$$

for all $(a,g) \in M \rtimes G$.

3. A general attack on the scheme

We will attack a stronger scheme, where only one of the groups A or B is made public. Without loss of generality, we may assume that A is known. A is generated by a given k-element subset. Let $(a_1,s_1),\dots,(a_k,s_k) \in M \rtimes G$ be the given generators of A. Let $S = \{s_1,\dots,s_k\}$. $S^{\pm 1}$ denotes the symmetrized generating set $\{s_1,\dots,s_k,s_1^{-1},\dots,s_k^{-1}\}$.

3.1. **Assumptions.**

3.1.1. *Distributions and complexity.* Alice and Bob make their choices according to certain distributions. Whenever we mention a *probability*, it is meant with respect to the relevant distribution. All assertions made here must hold "with significant probability" and the generation of elements must be possible within the available computational power. We will quantify our statements later.

Assumption 1. It is possible to generate an element $(\alpha,1) \in A$ with $\alpha \neq 1$.

Assumption 1 is equivalent to the possibility of generating $(\alpha,g) \in A$ such that the order o of g in G is smaller than the order of (α,g) in $M \rtimes G$. Indeed, in this case $(\alpha,g)^o$ is as required.

Assumption 2. N is a subgroup of $\mathrm{GL}_n(\mathbb{F})$ for some field $\mathbb{F}$ and some n.

We do not make any assumption on the field $\mathbb{F}$.

Alice generates an element $(a,g) \in A$, and in particular she generates g in the subgroup of G generated by S.

Assumption 3. Given $g \in \langle S\rangle$, g can be explicitly expressed as a product of elements of $S^{\pm 1}$.

3.2. **The attack.**

3.2.1. *First phase: Finding d and $\varphi(b)$ up to a scalar.* C, D commute elementwise. Use Assumption 1 to get a nontrivial $(\alpha, 1) \in A$. By $\star$-commutativity of $(\alpha, 1)$ with (b, h), we have that $\varphi(\alpha)\varphi({}^1b) = \varphi(\alpha)\varphi(b) = \varphi(b)\varphi({}^h\alpha)$, where only $\varphi(b)$ is unknown. Writing $\nu_1 = \varphi(\alpha), \nu_2 = \varphi({}^h\alpha)$, we summarize this by

$$\nu_1\varphi(b) = \varphi(b)\nu_2 \tag{4}$$

Now, $q = d\varphi(b)$ is a part of the transmitted information. Substituting $\varphi(b) = d^{-1}q$ in Equation (4), we obtain $\nu_1 d^{-1}q = d^{-1}q\nu_2$, and therefore

$$d\nu_1 = \nu_3 d$$

where $\nu_3 = q\nu_2 q^{-1}$. Now, choose a generic element $\gamma \in C$. Then

$$d\gamma = \gamma d.$$

We obtain $2n^2$ equations on the n^2 entries of d. As standard distributions were used to generate the keys, we expect that with overwhelming probability, the solution space will be one-dimensional. (As this is a homogeneous equation and the matrices are invertible, the solution space cannot be zero-dimensional.) If it is accidentally not, we can generate more equations in the same manner.

Thus, we have found xd for some unknown scalar $x \in \mathbb{F}$. Now use our knowledge of $q = d\varphi(b)$ to compute

$$(xd)^{-1}q = \frac{1}{x}\, d^{-1}q = \frac{1}{x}\, \varphi(b).$$

In summary: We know xd and $x^{-1}\varphi(b)$, for some unknown scalar $x \in \mathbb{F}$.

3.2.2. *Second phase: Generating elements with a prescribed G-coordinate and extracting the key.* Using Assumption 3, find $i_1, \dots, i_\ell \in \{1, \dots, k\}$ and $\epsilon_1, \dots, \epsilon_\ell \in \{1, -1\}$ such that

$$g = s_{i_1}^{\epsilon_1} \cdots s_{i_\ell}^{\epsilon_\ell}.$$

Compute

$$(\alpha, g) = (a_{i_1}, s_{i_1})^{\epsilon_1} \circ \cdots \circ (a_{i_d}, s_{i_\ell})^{\epsilon_\ell} \in A.$$

α may or may not be equal to a.

Remark 4. If M is generated as a monoid, the expression in Assumption 3 should be as a product of elements of S. In the cases discussed later in this paper, $G = S_n$ and the methods of Section 5 can be adjusted to obtain positive expressions (Remark 7).

By $\star$-commutativity of (α, g) and (b, h), $\varphi(b)\varphi({}^h\alpha) = \varphi(\alpha)\varphi({}^gb)$, and thus we can compute

$$x^{-1}\varphi({}^gb) = \varphi(\alpha)^{-1}(x^{-1}\varphi(b))\varphi({}^h\alpha).$$

We are now in a position to compute the secret part of the shared key, using Equation (4):

$$(xd)p(x^{-1}\varphi({}^gb)) = dp\varphi({}^gb).$$

The attack is complete.

4. Cryptanalysis of CBKAP

Anshel, Anshel, Goldfeld, and Lemieux propose in [2] an efficient concrete realization which they name *Colored Burau Key Agreement Protocol (CBKAP)*. We give the details, and then describe how our cryptanalysis applies in this case.

4.1. CBKAP.

CBKAP is the Eraser Key Agreement scheme in the following particular case. Fix a positive integers n and r, and a prime number p.

(1) $G = S_n$, the symmetric group on the n symbols $\{1, \ldots, n\}$. S_n acts on $\mathrm{GL}_n(\mathbb{F}_p(t_1, \ldots, t_n))$ by permuting the variables $\{t_1, \ldots, t_n\}$.
(2) $N = \mathrm{GL}_n(\mathbb{F}_p)$.
(3) $M \rtimes S_n$ is the subgroup of $\mathrm{GL}_n(\mathbb{F}_p(t_1, \ldots, t_n)) \rtimes S_n$, generated by (x_1, s_1), $\ldots, (x_{n-1}, s_{n-1})$, where s_i is the transposition $(i, i+1)$, and

$$x_1 = \begin{pmatrix} -t_1 & 1 & & \\ 0 & 1 & & \\ & & \ddots & \\ & & & 1 \end{pmatrix}; \quad x_i = \begin{pmatrix} 1 & & & & & & \\ & \ddots & & & & & \\ & & 1 & 0 & 0 & & \\ & & t_i & -t_i & 1 & & \\ & & 0 & 0 & 1 & & \\ & & & & & \ddots & \\ & & & & & & 1 \end{pmatrix}$$

for $i = 2, \ldots, n-1$. Only the ith row of x_i differs from the corresponding row of the identity matrix. The *colored Burau group* $M \rtimes G$ is a representation of Artin's braid group B_n, determined by mapping each Artin generator σ_i to (x_i, s_i), $i = 1, \ldots, n-1$.
(4) $\varphi : M \to \mathrm{GL}_n(\mathbb{F}_p)$ is the evaluation map sending each variable t_i to a fixed element $\tau_i \in \mathbb{F}_p$.
(5) $C = D = \mathbb{F}_p(\kappa)$ is the group of matrices of the form

$$\ell_1 \kappa^{j_1} + \cdots + \ell_r \kappa^{j_r},$$

with $\kappa \in \mathrm{GL}_n(\mathbb{F}_p)$ of order $p^n - 1$, $\ell_1, \ldots, \ell_r \in \mathbb{F}_p$, and $j_1, \ldots, j_r \in \mathbb{Z}$.

Commuting subgroups of $M \rtimes G$ are chosen once, by a trusted party, as follows:

(1) Fix $I_1, I_2 \subseteq \{1, \ldots, n-1\}$ such that for all $i \in I_1$ and $j \in I_2$, $|i - j| \geq 2$. $|I_1|$ and $|I_2|$ are both $\leq n/2$.
(2) Define $L = \langle \sigma_i : i \in I_1 \rangle$ and $U = \langle \sigma_j : j \in I_2 \rangle$, subgroups of B_n generated by Artin generators.
(3) L and U commute element-wise. Add to both groups the central element Δ^2 of B_n.
(4) Choose a random $z \in B_n$.
(5) Choose $w_1, \ldots, w_k \in zLz^{-1}, v_1, \ldots, v_k \in zUz^{-1}$, each a product of t-many generators. Transform them into Garside left normal form, and remove all even powers of Δ. Reuse the names $w_1, \ldots, w_k, v_1, \ldots, v_k$ for the resulting braids.

(6) Let $\rho : B_n \to M \rtimes S_n$ be the colored Burau representation function. A, B are the subgroups of $\rho(zLz^{-1}), \rho(zUz^{-1})$ generated by $\rho(w_1), \dots, \rho(w_k)$, and by $\rho(v_1), \dots, \rho(v_k)$, respectively.
(7) $w_1, \dots, w_k, v_1, \dots, v_k$ are made public.

Recall that to carry out our attack, it suffices to assume that one set of generators, $\rho(w_1), \dots, \rho(w_k)$ or $\rho(v_1), \dots, \rho(v_k)$, is given.

4.2. **The attack.** Assumption 2, that N is a subgroup of $\mathrm{GL}_n(\mathbb{F})$ for some field $\mathbb{F}$, is a part of the definition of CBKAP. We consider the remaining ones. As the distribution used in CBKAP are not specified in [2], we assume standard distributions in all of our attacks: Whenever, in the above descriptions, a product of a fixed number of elements of a set is required, we chose all of the elements independently and uniformly at random from that set. We then proceeded as instructed (for example, by reducing the powers of Δ^2 as mentioned above).

4.2.1. *Regarding Assumption 1.* This assumption amounted to: It is possible to generate, efficiently, an element $(\alpha, \sigma) \in A$ such that the order o of σ is smaller than that of (α, σ).

In the notation of Section 4.1, $\{i, i+1 : i \in I_1\}$ decomposes to a family $\mathcal{I}$ of maximal intervals $[i, \ell] = \{i, i+1, \dots, \ell\}$, and $\sum_{[i,\ell]\in\mathcal{I}} \ell - i + 1 \leq n/2$. Now

$$U = \langle \Delta^2 \rangle \oplus \bigoplus_{[i,\ell]\in I} B_{\ell-i+1}.$$

Each considered s is a permutation induced by the braid $\Delta^{2m} zwz^{-1}$ with $w \in L$. Let $\pi : B_n \to S_n$ be the canonical homomorphism. Then

$$s = \pi(\Delta^{2m} zwz^{-1}) = \pi(\Delta^2)^m \pi(z)\pi(w)\pi(z)^{-1} = \pi(z)\pi(w)\pi(z)^{-1},$$

is conjugate to $\pi(w)$. On each component, this is a product of many random transpositions, and is therefore an almost uniformly-random permutation on that component. We therefore have the following:

(1) $U/\langle \Delta^2 \rangle$ decomposes into a direct sum of braid groups, whose indices do not sum up to more than $n/2$.
(2) $\pi(U)$ decomposes into a direct sum of symmetric groups, whose indices do not sum up to more than $n/2$.
(3) For generic $(a, s) \in A$, $\pi(z)^{-1} s \pi(z)$ is generic on each part of the mentioned decomposition.

The probability that the order of a random permutation in S_n is $\leq n$ is $O(1/\sqrt[4]{n})$ [5]. Thus, we can find an element $(a, s) \in A$ with s of order $\leq n$ by generating (roughly $\sqrt[4]{n}$) elements $(a, s) \in A$, until the order of s is as required.

On the other hand, the element (a, s) is a representation of an element of the braid group, which is known to be torsion-free [14]. While the representation

used here may be unfaithful,[1] it is very unlikely that (a, s) could have finite order.

The remainder of this paper is dedicated to Assumption 3.

5. Membership search in generic permutation groups

For the second phase of our attack, it suffices to find a short expression of a given permutation in terms of given random permutations. Much work was carried out on this topic, by Babai, Beals, Hetyei, Hayes, Kantor, Lubotsky, Seress, and others (see [6, 4, 5] and references therein). Our approach is a heuristic shortcut for some of the ideas presented in these works. It performs surprisingly well on random instances of the problem.

Problem 5. *Given random $s_1, \dots, s_k \in S_n$ and $s \in \langle s_1, \dots, s_k \rangle$, express s as a short product of elements from $\{s_1, \dots, s_k\}^{\pm 1}$.*

In Problem 5, *short* could mean of polynomial length, or of length manageable by the given computational power as explained above. In any case, the length is the number of letters in the expression, and not the length of a compressed version of the expression. This limitation comes from the intended application, where we actually need to perform one $\star$ multiplication for each letter in the word. If the word is too long (e.g., of the form $a^{(2^{64})}$ for a single generator a), this becomes infeasible.[2]

For concrete generators, Problem 5 is well known, and in similar form occurs in the analysis of the Rubik's cube and other puzzles. The best known heuristics for solving it in these cases are based on Minkwitz's algorithms [13], and are incapable of managing Problem 5 for random $s_1, \dots, s_k \in S_n$ where n is large (say, $n \geq 128$), as experiments show.

A classical result of Dixon [7] tells that two random elements of S_n, almost always generates A_n (if all generators are even permutations) or S_n (otherwise). Babai proved that getting A_n or S_n happens in probability $1 - 1/n + O(1/n^2)$ [3]. Moreover, experiments show that this probability is very close to $1 - 1/n$ even for small n, i.e., the $O(1/n^2)$ is negligible also for small n. In particular, the probability that k random permutations do not generate A_n or S_n is (over-estimated by) at most $n^{-k/2}$, which is small for large n and negligible for large k.

Given that we obtain A_n or S_n, the probability of the former case is 2^{-k}. However, since $k = 2$ is of classical interest, we do not neglect this case.

Thus, for randomly chosen permutations Problem 5 reduces (with a small loss in probability) to the following one.

[1] It is open whether the colored Burau representation is faithful, even without reduction of the integers modulo p.

[2] The natural algorithm of repeated squaring does not help in the mentioned example: If a is a braid or its colored Burau representation, then each squaring makes a more complicated and the computation quickly becomes infeasible. The $\star$ operation avoids this problem, but does not admit an efficient analogue of squaring.

Problem 6.

(1) *Given random $s, s_1, \ldots, s_k \in A_n$, express s as a short product of elements from $\{s_1, \ldots, s_k\}^{\pm 1}$.*
(2) *Given random $s, s_1, \ldots, s_k \in S_n$ with some $s_i \notin A_n$, express s as a short product of elements from $\{s_1, \ldots, s_k\}^{\pm 1}$.*

A solution of Problem 6(1) implies a solution of Problem 6(2): Let $I = \{i : s_i \notin A_n\}$. $I \neq \emptyset$. Fix $i_0 \in I$, and for each $i \in I$, replace the generator s_i with the generator $s_{i_0}s_i \in A_n$. Then $\{s_{i_0}s_i : i \in I\} \cup \{s_i : i \notin I\}$ is a set of k nearly random elements of A_n (cf. [5]). If $s \in A_n$, use (1) to obtain a short expression of s in terms of the new generators. This gives an expression in the original generators of at most double length. Otherwise, $s_{i_0}s \in A_n$ and its expression gives an expression of s in terms of the original generators.

Thus, in principle one may restrict attention to Problem 6(1). However, we do not take this approach, since we want to make use of transpositions when we can.

5.1. **The algorithm.**

5.1.1. *Conventions.*

(1) During the algorithm's execution, the expressions of some of the computed permutations in terms of the original generators should be stored. We do not write this explicitly.
(2) The statement *for each* $\tau \in \langle S \rangle$ means that the elements of $\langle S \rangle$ are considered one at a time, by first considering the elements of $S^{\pm 1}$, then all (free-reduced) products of two elements from $S^{\pm 1}$, etc. (a breadth-first search), until an *end* statement is encountered.
(3) For $s \in (S^{\pm 1})^*$, $\mathrm{len}(s)$ denotes the length of s as a free-reduced word. s is identified in the usual way with the permutation which is the product of the letters in s.
(4) For $s \in S_n$, $\deg(s) = |\{k : s(k) \neq k\}|$.

We are now ready to describe the steps of our algorithm. We do not consider the question of optimal values for the parameters and other optimizations. This is left for future investigation.

Input: $G = S_n$ or A_n; generators $s_1, \ldots, s_k$ of G; $s \in G$.

Initialization: $\ell = n$;

$$c = \begin{cases} 2 & G = S_n \\ 3 & G = A_n \end{cases}$$

Step 1: *Find a short c-cycle in $\langle s_1, \ldots, s_k \rangle$.*

$t \leftarrow 0$.

For each $\tau \in \langle s_1, \ldots, s_k \rangle$:

  For each $m = 1, \ldots, \ell$:

    If $\deg(\tau^m) = c$:

$\mu = \tau^m$;
End Step 1.

The result μ of Step 1 is forwarded to the next step.

Step 2: *Find short expressions for additional c-cycles.*

Case $c = 2$**:**

For each $\tau \in \langle s_1, \ldots, s_k \rangle$:
$\pi \leftarrow \tau^{-1}\mu\tau$.
If π was not encountered before, store it.
If enough 2-cycles were found to present s by a short product of these, end Step 2.

Case $c = 3$**:**

If $s \notin A_n$:
Choose $s_i \in \{s_1, \ldots, s_k\}$ such that $s_i \notin A_n$;
$\sigma \leftarrow s_i s$.
Otherwise, $\sigma \leftarrow s$.

For each $\tau \in \langle s_1, \ldots, s_k \rangle$:
$\pi \leftarrow \tau^{-1}\mu\tau$.
If π was not encountered before, store it.
If enough 3-cycles were found to present σ by a short product of these, End Step 2.

Final step: *Find a short expression for s.*

Present s (or σ) as a product of the found cycles. Use the expressions of these cycles to get an expression of s in terms of the original generators.

Remark 7 (Positive expressions). If s belongs to the *monoid* generated by $\{s_1, \ldots, s_k\}$, we can adjust our algorithm to obtain a positive expression of s: Use only Step 1 (many times) to generate enough c-cycles to present s, where in this step, consider only words $\tau \in S^*$. This algorithm is more time consuming, but should still be successful in such scenarios. We do not pursue this direction here, since in CBKAP all involved algebraic objects are groups.

Remark 8 (Applicability to CBKAP). In CBKAP, G typically has the form $\pi^{-1}H\pi \leq S_n$, where $\pi \in S_n$, H is $S_{n/2}$ or $A_{n/2}$, and H is embedded in S_n in a natural way (supported by the $n/2$ higher indices). The conjugation is just relabelling of the indices $1, \ldots, n$. Thus, the algorithm applies without change to this case either. Modifications of the algorithm can be made, that will make it applicable to any (conjugation of) direct sum of groups of the form A_n or S_n.

6. Analysis of the generic membership search algorithm

6.1. **An idealized model.** For the heuristic estimations throughout this section, we make an optimistic assumption, whose consequences we verify experimentally later. This assumption is similar to one which was proved in [5]: For

random $\sigma, \tau \in S_n$, the lengths of the first cycles of $\sigma, \sigma\tau, \sigma\tau^2, \dots, \sigma\tau^\ell$ are pairwise nearly independent for $\ell \le n^{(7/32-o(1))\log n}$. However, a complete proof of our assumption would make a substantial breakthrough, since the state of the art provable algorithms, which can be deduced from [5] and [4], are more complicated, and their running time is $\Theta(n^7 \log n)$.

Assumption 9 (Near independence of enumerated elements). Let $k \ge 2$. For random, independently chosen $s_1, \dots, s_k \in S_n$, list the elements of $\langle s_1, \dots, s_n \rangle$ by first listing the elements of $\{s_1, \dots, s_k, s_1^{-1}, \dots, s_k^{-1}\}$, then all products of two elements from $\{s_1, \dots, s_k\}^{\pm 1}$ (which were not already listed), etc., to generate a sequence of desired length M.

We assume that for some non-negligible positive $\alpha \le 1$ (α may depend on n, but should not decrease quickly), the generated sequence contains a subsequence of αM elements, which looks (for the purposes of our analysis) like a sequence of αM random, independently chosen, elements of S_n. We call α the *density factor* for breadth-first search.

Assumption 9 is clearly true when $k \ge M$, but we usually apply it in cases where k is much smaller than M. In such cases, the density α cannot be 1, since e.g. the beginning of the sequence $s_1, \dots, s_k, s_1^{-1}, \dots, s_k^{-1}$ does not look random, even for some of our purposes. For simplicity, we carry out the analysis as if $\alpha = 1$. We name this model the *idealized model*. This means that actually, the resulting estimations on the required number of listed permutations should be multiplied by $\alpha^{-1} > 1$.

6.2. **Step 1.** The following terminology and lemma will make the proof of the subsequent theorem shorter. The *cycle structure* of a permutation $s \in S_n$ is the sequence $(n_1, n_2, \dots)$ of lengths of cycles of s which are not fixed points. Let $\sigma^n_{(n_1,\dots,n_k)}$ denote the number of elements of S_n with cycle structure $(n_1, \dots, n_k)$.

Lemma 10. $\sigma^n_{(n_1,\dots,n_k)} = \frac{n!}{(n-(n_1+\cdots+n_k))!\cdot n_1\cdots n_k}$.

Proof. First choose the $n_1 + \cdots + n_k$ elements which will occupy the cycles and consider all their permutations, and then divide out cyclic rotation equivalence, to get

$$\binom{n}{n_1+\cdots+n_k} \cdot (n_1+\cdots+n_k)! \cdot \frac{1}{n_1\cdots n_k}.$$

This is clearly equal to $\sigma^n_{(n_1,\dots,n_k)}$. $\square$

Proposition 11. *Let c be 2 if $G = S_n$, and 3 if $G = A_n$. For random $\tau \in G$, the probability that there is $d \in \{1, \dots, n\}$ such that τ^d is a c-cycle is greater than $1/cn$.*

Proof. In fact, we give better bounds for most values of n. We consider the probabilities to have cycle structures $(n-d, c)$ or $(n-d, e, c)$ for appropriate d, such that if τ has such a cycle structure, then τ^{n-d} is a c-cycle. The restrictions on the cycle structures are as follows.

(1) c does not divide $n-d$; and
(2) e divides $n-d$ (in the case $(n-d,e,c)$).

In the case $G = A_n$, we also must have that the cycle structure is possible in A_n:

(3) $n-d$ is odd (in the case $(n-d,3)$);
(4) $n-d+e$ is even (in the case $(n-d,e,3)$).

Assuming these restrictions, we compute the probabilities of these cycle structures using Lemma 10. In S_n, the probability for $(n-d,2)$ is

$$\frac{1}{|S_n|}\cdot\sigma^n_{(n-d,2)} = \frac{1}{(d-2)!\cdot(n-d)\cdot 2} > \frac{1}{(d-2)!\cdot 2n}.$$

In A_n, the probabilities for $(n-d,3)$ and $(n-d,e,3)$ are

$$\begin{aligned}\frac{1}{|A_n|}\cdot\sigma^n_{(n-d,3)} &= \frac{2}{(d-3)!\cdot(n-d)\cdot 3} > \frac{2}{(d-3)!\cdot 3n},\\ \frac{1}{|A_n|}\cdot\sigma^n_{(n-d,e,3)} &= \frac{2}{(d-e-3)!\cdot(n-d)\cdot e\cdot 3} > \frac{2}{(d-e-3)!\cdot 3en},\end{aligned}$$

respectively. We now consider some possible cycle structure, and describe the restrictions they pose on n and their probabilities. Let $m=2$ if $G=S_n$, and 6 if $G=A_n$. We write the restriction on n as $n \bmod m$.

Group	$n \bmod m$	Cycle structure	Prob.	Accumulated prob.
S_n	0	$(n-3,2)$	$1/2n$	
		$(n-5,2)$	$1/6n$	$7/12n$
	1	$(n-2,2)$	$1/2n$	
		$(n-4,2)$	$1/4n$	$3/4n$
A_n	0	$(n-5,3)$	$1/3n$	$1/3n$
	1	$(n-5,2,3)$	$1/3n$	
		$(n-6,3)$	$1/9n$	$4/9n$
	2	$(n-3,3)$	$2/3n$	
		$(n-6,2,3)$	$1/3n$	$1/n$
	3	$(n-4,3)$	$2/3n$	
		$(n-5,2,3)$	$1/3n$	
		$(n-7,2,3)$	$1/6n$	$7/6n$
	4	$(n-3,3)$	$2/3n$	
		$(n-5,3)$	$1/3n$	
		$(n-6,2,3)$	$1/3n$	$4/3n$
	5	$(n-4,3)$	$2/3n$	
		$(n-6,3)$	$1/9n$	
		$(n-7,2,3)$	$1/6n$	$17/18n$

This completes the proof. □

Corollary 12. *Let c be 2 if $G=S_n$, and 3 if $G=A_n$. Execute Step 1 with random elements $\tau\in G$ instead of the enumerated ones. The probability that it does not end before considering λn permutations is smaller than $e^{-\lambda/c}$.*

Proof. By Proposition 11, the probability of not obtaining a c-cycle for λn randomly chosen $\tau \in G$ is at most

$$\left(1 - \frac{1}{cn}\right)^{\lambda n} = \left(\left(1 - \frac{1}{cn}\right)^{cn}\right)^{\frac{\lambda}{c}} < (e^{-1})^{\frac{\lambda}{c}} = e^{-\frac{\lambda}{c}}. \quad \square$$

Example 13. Let c be 2 if $G = S_n$, and 3 if $G = A_n$, and $\lambda = c\lambda_0 \log n$ for some constant λ_0. Then the probability in Proposition 12 is smaller than

$$e^{-\frac{c\lambda_0 \log n}{c}} = n^{-\lambda_0}.$$

Corollary 14. *Let c be 2 if $G = S_n$, and 3 if $G = A_n$. Consider Step 1 in the idealized model. The average number of τ considered in this step is smaller than cn.* □

6.3. **Step 2.** We consider the most simple interpretation for "enough c-cycles were found to present s by a short product": Present s as a product of at most $n/(c-1)$ c-cycles in some canonical way. Then repeat Step 2 until all these c-cycles were found.

Proposition 15. *Execute Step 2 with random elements $\tau \in G$ instead of the enumerated ones. Let $c = 2$ if $G = S_n$, and 3 if $G = A_n$. The average number of elements considered in this step is smaller than $(n^c/c) \cdot (\log n + 2)$.*

Proof. Each conjugation of a c-cycle by a random permutation gives a random c-cycle. Let σ_c^n be the number of c-cycles in S_n. $\sigma_2^n = n(n-1)/2$, and $\sigma_2^n = n(n-1)(n-2)/3$. In any case, $\sigma_c^n < n^c/c$.

To obtain all c-cycles in a prescribed list of k out of N elements, we wait on average: N/k steps to obtain the first element, $N/(k-1)$ steps to obtain the second element, etc. Now,

$$\sum_{i=k}^{1} \frac{N}{i} = N \sum_{i=1}^{k} \frac{1}{i} = N \cdot H_k,$$

where $H_k < \log k + 2$ is the kth Harmonic number.

In our case, k is the number of c-cycles in a canonical decomposition of a permutation, and thus $k < n$, and N is the number of c-cycles in S_n, and therefore $N < n^c/c$. Thus,

$$NH_k < \frac{n^c}{c}(2 + \log n). \quad \square$$

Corollary 16. *The average running time of the generic membership search algorithm, in the idealized model, is $O(n^3 \log n)$ if $G = S_n$, and $O(n^4 \log n)$ if $G = A_n$.*

Proof. Let $c = 2$ if $G = S_n$, and 3 if $G = A_n$. Step 2 consumes most of the time, and requires by Proposition 15 $O(n^c \log n)$ operations on permutations. Each operation on permutations requires $O(n)$ elementary operations. Together, we have $O(n^{c+1} \log n)$ elementary operations. □

The constants in the estimations of Corollary 16 are not big, as can be seen by inspection of Step 2.

6.4. **The expression's length.** Using Corollary 14 and Proposition 15, we can derive a rough upper bound on the *average* length of the expression provided by the generic membership search algorithm, assuming that reality is not far from the idealized model (we verify this experimentally below).

By Corollary 14, Step 1 uses on average at most cn permutations until finding a good one τ. If τ is the cn-th permutation in our breadth-first enumeration of $\langle s_1, \dots, s_k\rangle$, then its length d as a word in the generators satisfies

$$(2k-1)^{d-1} \le 2k(2k-1)^{d-2} \le cn$$

(there are $2k(2k-1)^{d-1}$ free-reduced words of length d). Thus

$$\operatorname{len}(\tau) \le \frac{\log(cn)}{\log(2k-1)} + 1.$$

Then, μ is at most an n-th power of τ. Thus on average,

$$\operatorname{len}(\mu) \le n\left(\frac{\log(cn)}{\log(2k-1)} + 1\right).$$

Then, by Proposition 15, about $(n^c/c)\log n$ permutations τ are generated, and the c-cycles $\tau^{-1}\mu\tau$ are computed. The average length of the generated τ-s is thus estimated by

$$\operatorname{len}(\tau) \le \frac{\log((n^c/c)\log n)}{\log(2k-1)} \approx \frac{c\log n}{\log(2k-1)}.$$

In the last approximation there is less need for precision, since in any case,

$$\operatorname{len}(\tau^{-1}\mu\tau) \le \operatorname{len}(\mu) + 2\operatorname{len}(\tau) \le n\left(\frac{\log(cn)}{\log(2k-1)} + 2\right).$$

Less than $n/(c-1)$ c-cycles are needed to present the given permutation. Thus, the average length of the resulting expression is bounded by

$$\frac{n}{c-1}\operatorname{len}(\tau^{-1}\mu\tau) \le \frac{n^2}{c-1}\left(\frac{\log(cn)}{\log(2k-1)} + 2\right).$$

Corollary 17. *Assuming that the idealized model is a good approximation to reality, the average length of the expression provided by the algorithm is not much more than*

$$\frac{n^2}{c-1}\left(\frac{\log(cn)}{\log(2k-1)} + 2\right) = O(n^2\log n),$$

where $c = 2$ if $G = S_n$, and $c = 3$ if $G = A_n$. □

Example 18. For $k = 2$ and $n = 2^m$, the estimation in Corollary 17 is:

$$\begin{aligned}
&\frac{2^{2m}}{c-1}\left(\frac{\log(c2^m)}{\log 3}+2\right)\approx \\
&\approx \quad \frac{2^{2m}}{c-1}\cdot\frac{\log(c2^m)}{\log 3}=\frac{2^{2m}\log_3(c2^m)}{c-1}\approx\frac{2^{2m}(m+1)\log_3 2}{c-1}\approx \\
&\approx \quad \frac{0.63}{c-1}\cdot(m+1)2^{2m},
\end{aligned}$$

up to a multiplicative factor close to 1.

7. Experimental results

Following are experimental results, which indicate to which extent our idealized model for estimating the performance of the generic membership search algorithm is correct. The most difficult case for this algorithm is where there are only $k = 2$ random generators s_1, s_2. Thus, all of our experiments were conducted for $k = 2$.

7.1. **Assumption 9: The density factor α.** We assumed that for random, independently chosen $s_1, \dots, s_k \in S_n$, when M elements of $\langle s_1, \dots, s_n\rangle$ are generated in a breadth-first manner, the resulting sequence of M elements is as good for our purposes as a sequence of αM random permutations, where α is not very small (though it may depend on n).

For various values of n, and for $G = S_n$ or A_n, we have calculated the average number of permutations considered in Step 1, in the idealized model (an implementation using random permutations), and in the real model. Table 1 presents the ratio between them, i.e., $1/\alpha$, obtained using 100 experiments. We observe that the density α decreases with n, but very slowly.

Table 1. Average value of $1/\alpha$

n	8	16	32	64	128	256
S_n	4.64	5.94	7.4	8.54	10.72	13.56
A_n	2.51	4.24	6.36	8.04	9.52	11.56

Even for $n = 256$, the real sequence need only be 12 times longer than the required sequence of independent random permutations. In the additional place where Assumption 9 was used, α was not far from 1/2.

7.2. **Conventions.** For each $n = 8, 16, 32, 64, 256$, we have conducted at least 1000 independent experiments altogether. As $k = 2$, in about 750 of these experiments $\langle s_1, s_2\rangle = S_n$, and in about 250, $\langle s_1, s_2\rangle = A_n$. The few cases where neither S_n nor A_n were generated were ignored.

Each of these many samples suggests a value for the considered parameter. We thus present the minimum, average, and maximum observed values (with the average boldfaced). We present the ratio between the actual value and the analytic estimation obtained in the previous section. The analytic estimations

can then be used to obtain the actual numbers. The ratios are quite good, and the analytic estimations are likely to be good for all values of n.

In all discussions below c is 2 if $G = S_n$, and 3 if $G = A_n$.

7.3. **Step 1.** The ratio between the number of permutations considered in Step 1 and the estimation cn in Corollary 14 is given in Table 2.

TABLE 2. Ratios for the number of permutations in Step 1.

n	8	16	32	64	128	256
S_n	0.06	0.03	0.02	0.01	0	0
	2.26	**2.53**	**3.47**	**5.05**	**5.4**	**8.55**
	112.13	45.88	25.22	102.81	52.62	77.31
A_n	0.04	0.02	0.01	0.01	0.01	0
	0.51	**0.51**	**1.35**	**1.28**	**2.56**	**1.9**
	7.63	4.15	15.5	7.73	12.65	17.5

7.4. **Step 2.** Table 3 gives the ratio between the number of permutations considered in Step 2, and the estimation $(n^c/c) \cdot (\log n + 2)$ in Proposition 15.

TABLE 3. Ratios for the number of permutations in Step 2.

n	8	16	32	64	128	256
S_n	0.11	0.23	0.44	0.87	0.59	0.48
	2.32	**1.65**	**1.59**	**1.36**	**1.53**	**1.35**
	261.78	20.38	13.17	8.38	5.41	4.12
A_n	0.06	0.19	0.21	0.21	0.56	0.19
	2.47	**1.3**	**1.17**	**1.25**	**1.23**	**1.33**
	144.71	17.83	5.57	5.12	5.11	2.04

The striking observation is that here, the density factor α is very good, and in fact *improves* with n. As Step 2 is the most time consuming part in our algorithm, this means that the overall running time is close to the one predicted in the idealized model.

7.5. **Length of the final expression.** The average length of the final expression of the given permutation is estimated in Corollary 17 to be, in the idealized model, below

$$\frac{n^2}{c-1}\left(\frac{\log(cn)}{\log(2k-1)} + 2\right).$$

Table 4 shows that this estimation is surprisingly good, and that in fact, the true resulting length is on average better than the theoretically estimated one.

The actual lengths of the expressions produced for the given permutations are given in Table 5. For clarity, the average lengths are rounded to the nearest integer.

Table 4. Ratios for the length of the final expression.

n	8	16	32	64	128	256
S_n	0.06	0.11	0.11	0.1	0.12	0.13
	0.26	**0.44**	**0.56**	**0.73**	**0.79**	**0.9**
	0.95	0.95	1.07	1.26	1.24	1.25
A_n	0.08	0.08	0.08	0.04	0.03	0.04
	0.31	**0.37**	**0.54**	**0.6**	**0.74**	**0.74**
	0.6	0.8	1.1	0.96	1.08	1.09

Table 5. Expression lengths using the generic membership search algorithm.

n	8	16	32	64	128	256
S_n	16	148	674	2603	14357	65063
	76	**580**	**3331**	**19078**	**91120**	**450450**
	275	1258	6344	33015	143344	631306
A_n	13	54	248	504	1640	9258
	48	**261**	**1698**	**8328**	**44739**	**195534**
	94	564	3454	13328	65354	286628

For comparison, we looked for expressions of permutations as short products, using GAP's Schreier-Sims based algorithm (division off stabilizer chains), which uses optimizations similar to Minkwitz's [13]. Here, we have 100 experiments for S_n and 100 experiments for A_n. Already for $n = 32$, the routines went out of memory in about 1/3 of the cases for A_n, and in about 2/3 of the cases for S_n. Thus, we also checked $n = 24$ and $n = 28$ ($n = 28$ is the largest index which the routines can handle well). The resulting lengths are shown in Table 6, where ∞ means "out of memory in too many cases".

Table 6. Expression lengths using Schreier-Sims methods.

n	8	16	24	28	32
S_n	5	102	432	1047	∞
	22	**255**	**8039**	**345272**	∞
	42	418	350846	32729135	∞
A_n	0	95	549	913	∞
	18	**238**	**4101**	**59721**	∞
	29	413	35447	4012292	∞

We can see that Schreier-Sims methods are better than ours only for small values of n. Also, note the large difference between the minimal and the maximal obtained lengths. Contrast this with the results in Table 5.

8. Possible fixes of the Algebraic Eraser and challenges

As we have demonstrated, no choice of the security parameters makes the Algebraic Eraser immune to the attack presented here, as long as the keys are generated by standard distributions.

A possible fix may be to change the group S into one whose elements do not have short expressions in terms of its generators. This may force the attacker to attack the original matrices (whose entries are Laurent polynomials in the variables t_i) directly, using linear algebraic methods similar to the ones presented here. It is not clear to what extent this can be done.

The most promising way to foil our attacks, at least on a small fraction of keys, may be to use very carefully designed distributions, which are far from standard ones. Following our attack, Dorian Goldfeld found a distribution for which the equations in phase 1 of the attack have a huge number of solutions, and not all of these solutions lead to the correct shared key. This may lead to a system resisting the type of attacks presented here.

Another option would be to work in semigroups, and use noninvertible matrices. This may foil the first phase of our attack.

The generic membership search algorithm is of interest beyond its applicability to the Algebraic Eraser. We have demonstrated, by an idealized analysis supported by experiments, that this algorithm easily solves instances with random permutations, in groups of index which is intractable when using previously known techniques like those in [13].

The most interesting direction of extending the present work seems to be a rigorous analysis, in the real model, of this algorithm. This would be a mathematical breakthrough, since the state of the art provable algorithms, despite being more sophisticated, have running time $\Theta(n^7 \log n)$, which is not practical for $n \geq 128$. Alexander Hulpke has informed us that our methods are similar to ones used for constructive recognition of S_n or A_n. This connection may be useful for the proposed analysis.

Finally, we point out that even without changes, our algorithm applies in many cases not treated here. Experiments of the full attack succeeded to extract the shared key correctly in *all* tested cases, including some in which the index of the generated subgroup of $S_{n/2}$ was greater than 2.

Acknowledgements. This research was partially supported by the Emmy Noether Research Institute for Mathematics and the Minerva Foundation of Germany. We thank Dorian Goldfeld, Stephen Miller, Ákos Seress, and Adi Shamir, for fruitful discussions. We also thank Alexander Hulpke and Stefan Kohl for useful information about GAP. Our full attack on the Algebraic Eraser was implemented using MAGMA [12].

References

[1] I. Anshel, M. Anshel and D. Goldfeld, *An algebraic method for public-key cryptography*, Mathematical Research Letters **6** (1999), 287–291.

[2] I. Anshel, M. Anshel, D. Goldfeld, and S. Lemieux, *Key Agreement, the Algebraic Eraser*$^{\mathrm{TM}}$*, and lightweight cryptography*, Contemporary Mathematics **418** (2006), 1–34.

[3] L. Babai, *The probability of generating the symmetric group*, Journal of Combinatorial Theory A **52** (1989), 148–153.

[4] L. Babai, R. Beals, and Á. Seress, *On the diameter of the symmetric group: polynomial bounds*, 15th ACM-SIAM SODA (2004), 1108–1112.

[5] L. Babai and T. Hayes, *Near-independence of permutations and an almost sure polynomial bound on the diameter of the symmetric group*, in: Proceedings of the Sixteenth Annual ACM-SIAM SODA, 2005, 1057–1066.

[6] L. Babai, G. Hetyei, W. Kantor, A. Lubotsky, and Á. Seress, *On the diameter of finite groups*, 31st IEEE FOCS, 1990, 857–865.

[7] J. Dixon, *The probability of generating the symmetric group*, Mathematische Zeitschrift **110** (1969), 199–205.

[8] The GAP Group, *GAP – Groups, Algorithms and Programming*, Version 4.4.10, 2007, `http://www.gap-system.org`

[9] D. Holt, B. Eick, and E. O'Brien, *Handbook of Computational Group Theory*, Chapmann & Hall/CRC, 2005.

[10] K. H. Ko, S. J. Lee, J. H. Cheon, J. W. Han, S. J. Kang and C. S. Park, *New Public-key Cryptosystem using Braid Groups*, CRYPTO 2000, Lecture Notes in Computer Science **1880** (2000), 166–183.

[11] C. Leedham-Green, C. Praeger, and L. Soicher, *Computing with group homomorphims*, Journal of Symbolic Computation **12** (1991), 527–532.

[12] Wieb Bosma, John Cannon, and Catherine Playoust, *The Magma algebra system, I: The user language*, Journal of Symbolic Computation **24** (1997), 235-265.

[13] T. Minkwitz, *An algorithm for solving the factorization problem in permutation groups*, Journal of Symbolic Computation **26** (1998), 89–95.

[14] K. Murasugi, *Seifert fibre spaces and braid groups*, Proceedings of the London Mathematical Society **44** (1982), 71–84.

[15] C. Sims, *Computational methods in the study of permutation groups*, in: J. Leech (ed.), **Computational Problems in Abstract Algebra**, 169–183, Pergamon, 1970.

(Arkadius Kalka, Mina Teicher) Department of Mathematics, Bar-Ilan University, Ramat-Gan 52900, Israel

(Boaz Tsaban) Department of Mathematics, Bar-Ilan University, Ramat-Gan 52900, Israel; and Department of Mathematics, Weizmann Institute of Science, Rehovot 76100, Israel.

E-mail address: `Arkadius.Kalka@rub.de`

E-mail address: `[teicher, tsaban]@math.biu.ac.il`

URL: `http://www.cs.biu.ac.il/~tsaban`